A HIDDEN SURFACE ALGORITHM FOR COMPUTER GENERATED HALFTONE PICTURES

John E. Warnock

Utah University

Prepared for:

Advanced Research Projects Agency
Rome Air Development Center

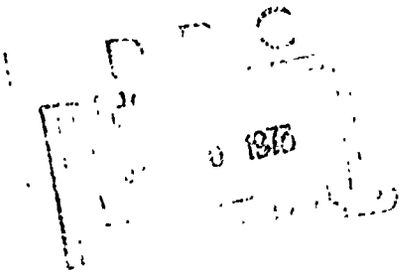
June 1969

RADC-TR-69-249
Technical Report
June 1969

A HIDDEN SURFACE ALGORITHM FOR COMPUTER GENERATED HALFTONE PICTURES

University of Utah

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 829



Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, New York

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Computer Science University of Utah Salt Lake City, Utah 84112	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

A HIDDEN SURFACE ALGORITHM FOR COMPUTER GENERATED HALFTONE PICTURES

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report

5. AUTHOR(S) (First name, middle initial, last name)
John E. Warnock

6. REPORT DATE June 1969	7a. TOTAL NO. OF PAGES 29	7b. NO. OF REFS 8

8a. CONTRACT OR GRANT NO. AF30(602)-4277 b. PROJECT NO ARPA Order No. 829 c. Program Code Number: 6D30 d.	9a. ORIGINATOR'S REPORT NUMBER(S) TR 4-15 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) RADC-TR-69-249

10. DISTRIBUTION STATEMENT

This document has been approved for public relase and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES Monitored by Rome Air Development Center (EMIIO) Griffiss Air Force Base, New York 13440	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency Wash DC 20301

13. ABSTRACT

The application of computer graphics to problem solving has increased over the past few years. The representation of data in the form of line drawings, graphs, charts, diagrams and line plots has been explored extensively. This paper addresses itself to some new techniques used to solve problems associated with extending the power of computer graphics to include black and white, and color shading. In particular it presents a new method for converting data describing three-dimensional objects into data that can be used to generate two-dimensional halftone images. It deals with some problems that arise in black and white, and color shading.

DD FORM 1 NOV 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Hidden-line						
Hidden-surface						
Grayscale						
Halftone						
Grapnics						
Color Shading'						
Color						
Raster						

A HIDDEN SURFACE ALGORITHM FOR COMPUTER GENERATED HALFTONE PICTURES

University of Utah

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by David A. Luther RADC (ISCE), GAFB NY 13440 under contract AF30(602)-4277.

FOREWARD

This interim report describes research accomplished by Computer Science of the University of Utah, Salt Lake City, Utah, for the Advanced Research Projects Agency, administered by Rome Air Development Center, Griffiss Air Force Base, New York under Contract AF30(602)-4277. Secondary report number is TR 4-15. Mr. David A. Luther (EMIIO) is the RADC Project Engineer.

This technical report has been reviewed and is approved.

David A Luther

Approved: DAVID A. LUTHER
Project Engineer

ACKNOWLEDGEMENTS

I express my appreciation to Dr. David C. Evans, Dr. Ivan E. Sutherland, and Dr. Dan Cohen for their great insight, help, and encouragement in the development of this work. The many hours they have spent with me in discussion have provided the intellectual stimulus required to carry on this research.

I am deeply indebted to Dr. Thomas G. Stockham, whose encouragement and friendship made this document possible. Without the time and effort given unselfishly by Dr. Stockham, this document would not exist.

Many thanks to Dr. William M. Newman whose suggestions concerning the manuscript were invaluable.

Thanks also to Linda Rae Buys for her help in typing the manuscript.

Lastly, I would like to thank my wife, Marva, for her unselfish love and help. Without her understanding and kindness, this research would not have been possible.

ABSTRACT

The application of computer graphics to problem solving has increased over the past few years. The representation of data in the form of line drawings, graphs, charts, diagrams and line plots has been explored extensively. This paper addresses itself to some new techniques used to solve problems associated with extending the power of computer graphics to include black and white, and color shading. In particular it presents a new method for converting data describing three-dimensional objects into data that can be used to generate two-dimensional halftone images. It deals with some problems that arise in black and white, and color shading.

TABLE OF CONTENTS

SECTION	PAGE
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
INTRODUCTION	1
A NEW PHILOSOPHY OF HIDDEN SURFACE ALGORITHMS	2
DATA STRUCTURE	8
VARIATIONS ON THE THEME	11
THE DISPLAY OF PICTURES	14
SHADING AND COLOR	17
CONCLUSION	23
BIBLIOGRAPHY	26
VITA	27

INTRODUCTION

The past few years have seen an enormous increase in the use of computer graphics. Nevertheless, the medium is still very restricted and primitive.[5] Present day equipment makes it impossible to achieve the pictorial realism that a graphic artist can attain. Coloring, shading, texture, and lighting effects are not available to the computer graphics user. With the present state of computer graphics in mind it is meaningful to ask what current technology can do to enhance the power of computer graphics. The aspects we would like to control may include intensity, color, and location of light sources; reflectance, surface texture, and coloring of the objects; and general illumination and atmospheric interference in the picture field. If these parameters can be controlled then computer graphics will offer a powerful tool for generating visual images.

This paper will address itself to some of the problems that arise in trying to attain the above goals. In particular it will deal with the problem of converting data that describes objects in three-dimensional space into data that can generate a two-dimensional picture representation of those objects. This problem is generally called the "Hidden Line" or "Hidden Surface" problem. The final section of this paper discusses the illumination, coloring and shading of picture representations of three-dimensional objects, and the display file structures that generate raster scan images on a display.

A NEW PHILOSOPHY OF HIDDEN SURFACE ALGORITHMS

The description of the philosophy is best given by describing the motivation behind it. Suppose I examine a picture of a table with pencils on it. I quickly determine that large areas on top of the table are open and therefore have little information content. I scan over these areas rearching out complex features such as a pencil. I dwell on a complex portion until I assimilate the information associated with it. From there I scan to other areas of the picture looking for additional complexity. In scanning the picture in this way I seem to spend little or no time on simple areas. Complex areas seem to present themselves to me as subproblems requiring a solution. I seem to reduce these problems into further subproblems until I either solve the subproblem or don't care anymore.

New algorithms for removing hidden surfaces which emulate the above process will now be discussed. Consider the view plane as a square picture. Either its contents are simple enough to process with some decision procedure or they are not sufficiently simple. If they are not, then we subdivide the square into four subsquares and ask the same question about the contents of the first subsquare as was asked about the parent square. This process is repeated recursively until one of two conditions holds with respect to a subsquare. If the contents of the square are simple enough to process, then data is put into a display file. If the square is as small or smaller than the desired resolution of the picture, then put a reference to the surface nearest to the observer into the display file.

The decision procedure that determines if subdivision of squares is necessary can be simple. Suppose a picture is of a set of planar polygons. We will generally subdivide a square unless one of two conditions holds. The projections of the polygons onto the view plane do not intersect with the square; or the square is surrounded by a projected polygon which is in front of other extended polygons within the square. Figure 1a illustrates what a view plane may look like if subdivided by the above process.

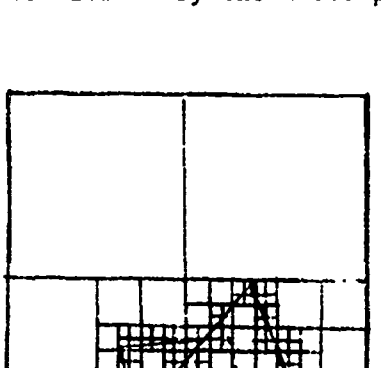


Figure 1a

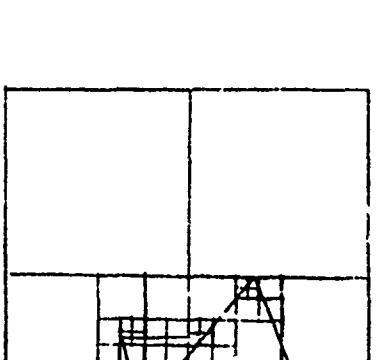


Figure 1b

This criteria for subdivision will find all resolution size squares along the visible edges and all resolution size squares along the visible intersections of planes. This is exactly what we want. It locates all the points along the visible edges and intersections of the objects.

The process of subdividing the picture can be thought of as a logarithmic search for points where there is a change in the characteristics of the picture. The process can also be thought of as a scheme

for finding locally simple geometric phenomena in portions of the view plane. Some observers characterize the algorithm as being what Floyd[3] calls "non-deterministic". Another point of view sees the algorithm as making a tree of goals. As a goal is considered, a decision is made about its difficulty. If it cannot be achieved with the methods at hand, the algorithm generates subgoals (four). The first subgoal is now considered. This process is repeated recursively until either success is reported at some subgoal or until success is reported by default (In our case when the resolution of the picture is reached). When success is reported, the next subgoal at the same level (the brother) is considered. If all subgoals of a given goal have reported success, then the goal itself reports success. This order of examining goals and their subgoals is called a "prefix" ordering of the tree. In any event the important thing is that a difficult problem can successfully be divided into smaller, easier subproblems. This approach has many programming and data structuring advantages.

Subdividing the picture in a prefix order allows the computing associated with each square to be reduced by saving information about what happened in the larger "parent" squares. It is useful to save information invariant under the operation of subdivision. Such information can be called "inherited". Taking advantage of inherited information can significantly reduce the computation associated with small squares.

Let us consider how inherited information might be used when a square and a polygon are disjoint. In this case the fact that the polygon and square are disjoint is inherited by all subsquares of that square, i.e., the subsquares are also disjoint with the polygon.

When inherited information given in this example is used, it is generally true that the number of planar polygons processed becomes smaller as the squares examined get smaller. Using this fact allows large amounts of data not relevant to the local problems to be shelved. Saving the level of refinement at which the data is shelved allows the reconsideration of that data when coming up out of the subdivision process. Data is reintroduced at the level at which it was shelved so that brother squares at that level may be correctly processed.

The decision procedure that determines if a square is to be subdivided is important. If the procedure is relatively simple, then many subsquares may be processed. If the procedure is complex, then a fewer squares may be processed and yet computation per square may be high. In our discussion of the hidden surface algorithm three decision procedures will be discussed. The first will be discussed now. The last two will be introduced under the heading "Variations on the Theme". Let us assume that the input to the algorithm is any collection of planar polygons in three-space. Also assume that a polygon k is given as a sequence $(P_{ki} = (X_{ki}, Y_{ki}, Z_{ki}) \mid i = 1,2 \ldots n)$ of n ordered planar points that form a directed closed path in three-space. Let σ be a map from three-space onto the view plane P such that $(P_{ki})\,\sigma = (P'_{ki})$, for all i. Let $d_k(u,v,w)$ be equal to the distance from the point (u,v,w) on the view plane to the point (x,y,z) on the plane of polygon k such that $(x,y,z)\,\sigma = (u,v,w)$, (see Figure 2). We can now describe a compact decision procedure that will remove hidden surfaces from the representation on P.

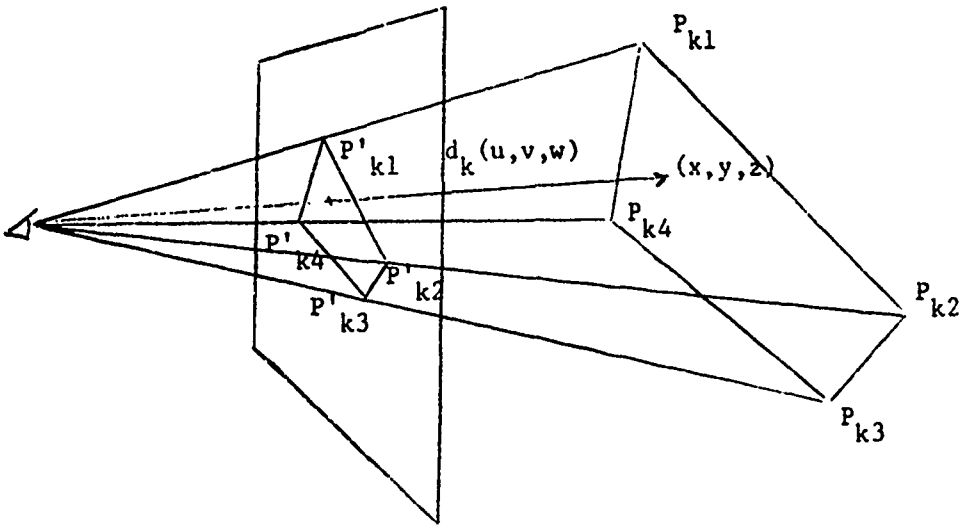


Figure 2

There are three basic relationships that have to be established between each polygon and the square being processed.

1.) The square is wholly inside the projected polygon.

2.) The square and the projected polygon intersect, or

3.) The square is wholly outside the projected polygon

(see Figures 3a, 3b, and 3c). Note that condition 1 and 2 may not be mutually exclusive (see Figure 3d).

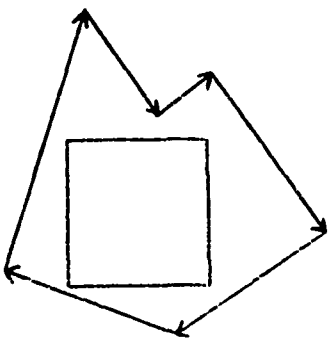


Figure 3a

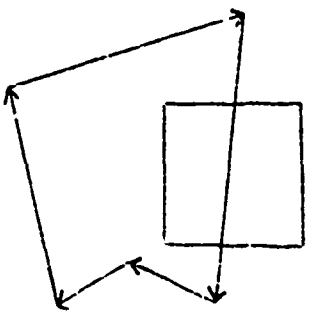


Figure 3b

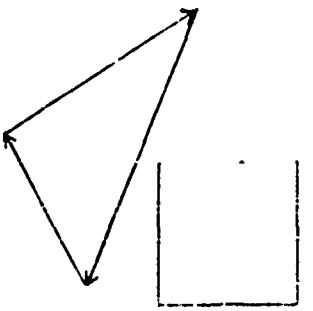


Figure 3c

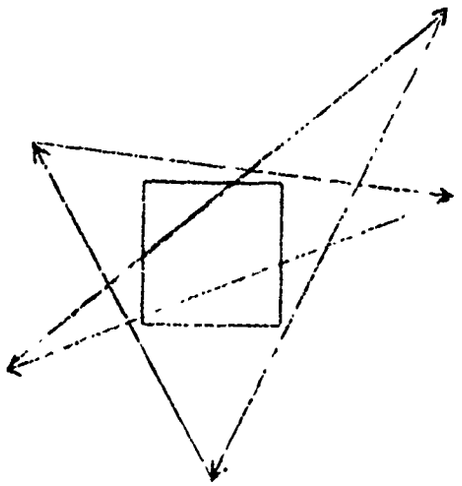


Figure 3d

To determine which of these relationships exists, tests are made on the vertices of the polygon to determine if any are in the square. If any vertex of the polygon is in the square, then condition 2 is true. The edges that comprise each polygon are then individually checked to see if any intersect the square. This is done by seeing if either of the segments that are the diagonals of the square intersect the given edge. If any edges lie within the square then although condition 1 might be true we will assume condition 2 is true, e.g. Figure 3d. If no vertices or edges are within the square, then one corner $(w_j, u_j, v_j,)$ of the square is checked to see if it is in the interior of the polygon. This is done by letting R be any ray in the view plane P eminating from (w_j, u_j, v_j). Now each edge of the polygon is checked against R to see if it crosses R. Also the direction in which edges cross is recorded. If the number of positive crossings equals the number of negative crossings, then the square is on the exterior of the polygon, (condition 3) otherwise, it is wholly inside (condition 1).

Once the relationships between the square and the polygons have

been determined, subdivision of the square will occur unless one of the following conditions is satisfied. Is there a polygon satisfying condition 1 that hides all other polygons? More precisely, does there exist a polygon m satisfying condition 1 such that $d_m(u_j, v_j, w_j) = \min |d_i(u_j, v_j, w_j)|$ for all $i \neq m$ such that polygon i satisfies condition 1 or 2| for j=1,2,3,4 where (u_j, v_j, w_j) are the corners of the square? If such a polygon exists, then subdivision does not occur. The other condition, where subdivision does not occur, is satisfied provided all polygons have condition 3 with the square, i.e., the square "sees" nothing.

Fortunately, in practice, we are able to simplify the tests for the relationships between polygons and squares. Let the view plane P be the X,Y plane and the projection be an orthogonal projection onto that plane. This assumption is not restrictive if the polygons are put through appropriate translation, rotation, clipping, and perspective transformations before projection. If the above special case is assumed, then $P'_{ki} = (X_{ki}, Y_{ki}, 0)$ and $d_k(x,y,0) = Ax + By + C$ where $Z = Ax + By + C$ is the transformed plane of the polygon in question. Also, if the ray used in checking for condition 1 is assumed to have zero slope, then computation is simplified.

The decision procedure just described will produce resolution size squares along visible edges and intersection of the polygons. If these are directly displayed then a line drawing of the object with hidden surfaces removed will result. See Figure 5a.

DATA STRUCTURE

The preceding discussion has mentioned nothing about the data

structures that are used in implementing the algorithm. We assume that the input to the algorithm is restricted to be a set of arbitrary planar polygons in three-space.

Polygons can be input in a variety of ways. They can be considered to be a sequence of points (x_i, y_i, z_i) in three-space that form a closed path. They can be considered to be an initial point (Xo,Yo,Zo) followed by increments (ΔXi,ΔYi,ΔZi) from the previous point to form a closed path. A third way to specify the polygons given to the algorithm is to give the x and y coordinates of the points defining the polygons plus the equations of the planes of the polygons in the form $Ax + By + Cz + D = 0$. Any of these input schemes or other more elaborate schemes are useful depending on the application.

Along with the points that make up the polygons the number and order of points in each polygon is important in the input. Other data relevant to the polygons are an array of pointers making the set of input polygons into a list. This list structure is important in dealing with the inherited information discussed earlier.

Another essential data structure is that which is output during execution of the algorithm. I call this the display file. The nature of the display file is determined by the decision procedure used to decide if subdivision of a square is necessary. If the decision procedure is the one described earlier, then the display file consists of the points (x,y) on the view plane that are either on visible boundaries of polygons or on visible intersections of polygons. Information associated with these points in the display file depends on the type of picture and shading to be produced. The

choice of a particular decision procedure is influenced by the capability of the display processor. Let us now explore these data structures in the context of the algorithm.

The list structure that we impose on the set of input polygons facilitates the exploitation of the inherited properties of the squares. For instance, the relationship between a given square and polygon can be one of three types. The square is wholly contained within the polygon. The square can contain an edge or vertex of the polygon. Or the square and polygon can be disjoint. The first and third of these conditions are inherited by all subsquares of a square. Suppose we treat the set of polygons as a list broken into three parts. The first part is all those polygons satsifying condition 1, the second part those satisfying condition 2, and the third those satisfying condition 3. Make the pointer to the first of the list of polygons point to the head of part 1. All polygons in part 1 surround any subsquare and therefore need not be checked for this condition. If a polygon in part 2 is found, upon checking, to surround the current subsquare, then this polygon can be inserted on the tail of part 1 for use in the next level of recursion. If any polygon in part 2 is found to be disjoint with the current subsquare, then this polygon can be inserted at the head of part 3 for use in the next level of recursion. Polygons in part 3 need not be examined at all. Conditions 1 and 3 are inherited by all subsquares of a given square. Condition 2 may not be. For this reason each polygon satisfying condition 2 must be checked with respect to each subsquare to see which of the three conditions holds. If we keep pointers to the end of part 1 and to the beginning of part 3 for each level in the recursion, then

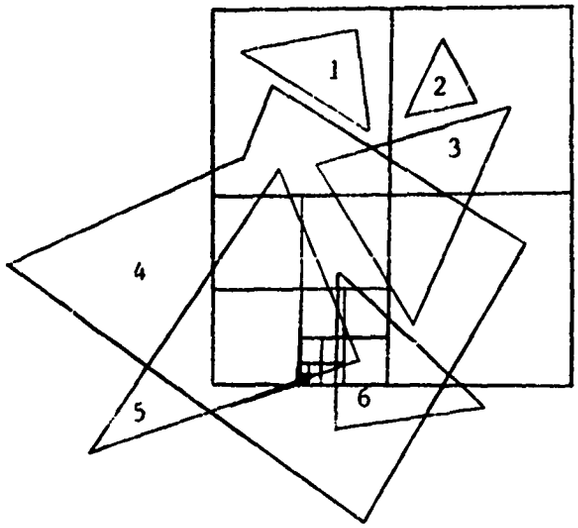
given any level we can determine what relationships the polygons have with the current square at that level. See Figure 4 for an example of a recursion and the list associated with it.

The list described above nicely structures some of the inherited information. Other information that is inherited by the squares is not easily handled by the list. An example of this occurs when a polygon has exactly one edge that intersects the square. Note that if this is the case then only one edge need be checked for intersection with any subsquares. This type of inherited information can be saved by tagging each polygon with the level at which the condition was first discovered plus the number of the edge that crosses the square. Other inherited information can be utilized when a polygon surrounding the current square is in front of other polygons. The polygons that are hidden in this way need not be examined in any subsquares. These polygons are flagged in a way similar to the one above. By saving more information with each polygon, time could conceivably be saved but storage costs may be high. For this reason additional information with each polygon has not been saved but instead has been recomputed when needed.

VARIATIONS ON THE THEME

It is pertinent at this time to explore what might be done to reduce the number of output squares. This question is related to the decision procedure that decides if a square is too complicated to handle and therefore must be subdivided.

One approach taken to cut down the number of output squares is to terminate the subdivision of squares whenever at most one line of a polygon or intersection of two polygons is contained in the square. If this decision procedure is used, then an example of the resulting



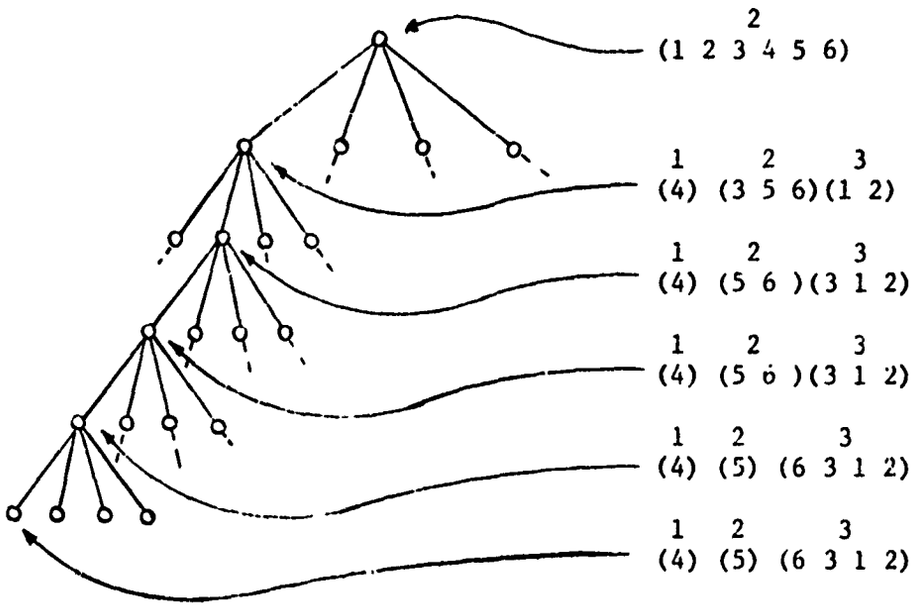


Figure 4

As subdivision of a square proceeds, the list of polygons reflects inherited information gathered. This is indicated by the above partial subdivision and by its associated tree. At each of the nodes the list is in at most 3 parts: those polygons surrounding the square (part 1); those intersecting the square (part 2) and those disjoint from the square (part 3).

mosaic of squares is given in Figure 1b. We can see from comparing Figure 1a and Figure 1b that this technique reduces the number of squares that need be examined in the algorithm. Since the additional logic required in the decision procedure is not excessive, the adoption of this procedure may result in faster execution.

Another approach that can be used to implement the general philosophy of the hidden surface algorithm dictates that squares may not be the most desirable geometric entity to subdivide. This approach might suggest that since scan lines are the ultimate output, scan lines should be the construct to be subdivided. Although there are no conceptual advantages to this method, data structuring and memory management schemes can be invoked that may produce savings.

The basic strategy of this approach assumes, as in the previous cases, that a collection of planar polygons are input to the algorithm. If a list is associated with the array of points of the polygons, then this list can reflect the order of the points such that the y coordinate of the points decreases as the list is followed. This implicit sorting of the points allows the edges of the polygons to be entered and deleted from a "current edge" list as they are germane to the computation of successive scan lines. An edge enters the list when the y coordinate of the scan line matches the greater of the y coordinates of the edge. The edge is deleted when the other y coordinate is matched by the y coordinate of the scan line. When an edge is introduced into the list several facts are stored with it. The polygon to which the edge belongs is stored. The initial x coordinate of the edge along with the change in x with respect to y is stored.

Under this approach the strategy for processing a given scan line coincides with the strategy for the entire picture in the previously described schemes. The scan line is recursively divided in half if its contents are too complicated to process. This technique of implementation allows the "inherited" information to be utilized to its full extent and also allows for a much simpler computation on the relationship between a segment of a scan line and a polygon. This method has the disadvantage, however, that it must go through the subdivision process for each scan line that intersects some polygon.

Up until now we have discussed three variations for the implementation of the hidden surface algorithm. The strategy that minimizes time in the context of the algorithm's philosophy can only be obtained by examining computation trade-offs within the algorithm. This paper will not dwell on these problems since the evidence we have collected does not conclusively indicate that one method is superior to another.

THE DISPLAY OF PICTURES

There is something common to all types of display files the algorithms may output. The information in the display file represents the visible geometric aspects of the picture. The way the file represents the visible geometry is a function of the decision procedure that allows the subdivision of squares. If the subdivision of a square continues when any visible boundaries are in the square, then the display file consists of the points along the visible boundaries and intersection of the polygons. In this case the display file is a set of points (x,y). If line representations of the picture are desired, then the action to be taken is quite simple. The set of (x,y)

points must be sorted into the order that corresponds to the way the raster sweeps the display, i.e., if the raster sweeps left to right, top to bottom, then sort the points x ascending within y descending. If the scan proceeds and each point in the scan is compared with the first point in the sorted display file until a match is found, then that point can be intensified. The scan can now be continued until a match with the next point in the display file is found. This process of matching and advancing the scan and display file is continued until the entire picture is produced (see Figure 5a). The above scheme is basically the same for all of the techniques that we have developed for converting the geometry of a picture into a raster scan image. In all these techniques the display file is sorted so that its elements are encountered in the same order as the raster sweeps the screen. If the geometry consists of points as in the above case, then the points are encountered in the same order as the scan sweeps those points. If the geometry consists of line segments as is the case if the second decision procedure is adopted, then line segments are ordered so they are encountered sequentially by the raster.

To build a picture from line segments the approach taken is more complicated. Once a line is encountered by the raster it is used in generating the picture until the line segment is exhausted by the scanning process. Memory is dedicated for storing the line segments that are currently pertinent to the generation of the display. When these line segments are encountered as the scan sweeps the display, they are inserted into memory. The segments are then generated from memory point by point until exhausted. They are then deleted. Since the algorithm cannot allow line segments to intersect, it is

seen that the updating of the segment from scan line to scan line can never alter their order in the memory for current lines. For this reason the memory in which the segments are stored can be thought of as a circular queue. In conjunction with the processing of segments, new segments are introduced into the queue as they are encountered by the raster. Information that is required with each line segment varies with the kind of shading desired. If one wants to display the lines themselves, then the following is needed: The initial point of each segment (x_0, y_0), the inverse of the slope $\Delta x/\Delta y$ and the last y coordinate of the segment y_L. The segment is introduced into the memory based on x_0 and y_0. It is deleted when a match on y_L occurs. The intermediate values are attained by updating x_0 with $\Delta x/\Delta y$. This scheme requires no searching and therefore is quite efficient.

If halftone shading of the polygons is desired with the first kind of display file, then the action to be taken is simple. When the polygons are passed to the hidden line routine, a shading function is computed for each polygon. This function is in terms of x and y and yields a brightness level for that polygon at a point (x,y). In the process of hidden line removal each display file entry (x,y) is tagged with a pointer to the polygon seen immediately to its right. If there is no polygon to the right then a null polygon is used. With this display file it is easy to produce a shaded picture. The display file is sorted x ascending within y descending. As each point is encountered by the scan, the shading function associated with the polygon is put into effect along the scan line until the next point is encountered by the scan and its shading function takes effect. This method allows a flexible use of the halftone display. By overlaying three shading functions correspondin

to the additive primary color components on a given polygon it is quite easy to achieve full color capabilities in the shading of the objects displayed. What these shading functions are and how they are achieved will be the next topic of discussion.

SHADING AND COLOR

The production of shaded pictures in black and white or "natural" color presents some very unusual problems. The variables of which the worker must be aware in the consideration of a model include the following:

a. The spectral components of the simulated or fictional source light.
b. The spectral filtering and scattering effects of the simulated or fictional atmosphere in which the scene resides.
c. The surface reflectance qualities of the simulated or fictional objects.
d. The absorption and reflectance properties of the various parts of the spectrum by the color of the object.
e. The distance and angle of the polygons with respect to the source light.
f. The nonlinearities in the circuitry to the display system.
g. The spectral components of the light emission of the display phosphor as a function of beam current.
h. The spectral filtering of the color separation filters used.
i. The color distortion of the photographic lens.
j. The sensitivity of the film to various areas of the spectrum.
k. The reflectance properties of the dyes used in the film.

In lieu of building a mathematical color model that will compensate

and account for all of the above variables, a description will be given of color and shading models that will yield acceptable results. These models have been constructed on empirical considerations and should in no way be regarded as models for colorimetry, optics, or psychophysics.

The shading models and color models that have been used will be discussed in the order in which they evolved.

The simplest approach to black and white shading is to ignore all aspects of lighting except the angle between a ray from the light source and the normal to the polygon under consideration. If this is done then the shade of a polygon can be given by $s = K|\cos \theta| + M$ here K and M adjust the magnitude of the numbers going to the display. θ is the angle between a ray from the source and the normal to the polygon. This shading rule produces acceptable real-looking results in most cases. Its drawback can be noticed when a picture of two overlapping separated parallel planes is made and the edge that overlaps cannot be detected.

The second type of shading model considers the shading of a point on a polygon to be a function of its distance from the light source. The function can be given by $s = \frac{K|\cos \quad|}{r + C} + M$. Here r is the distance of the point from the light source. K, C, and M should be adjusted to the values of r so that an acceptable range of numbers is fed to the display.

It may be said at this point that illumination decreases with r^2 instead of r, and therefore r^2 should be used. Practically speaking the only reason that distance is considered at all is to eliminate the problem of the parallel planes described in the previous scheme. The use of r^2 can change the light intensity so radically over a short distance that the resulting pictures look unnatural. In fact, depending on the

values of r, the use of $\sqrt{r}$ may produce better results than r. In any event, the second scheme of shading produces good black and white pictures (see Figure 5b).

To open the discussion about producing color, a straightforward extension of the above will be considered. For each of the additive primary colors (red, green, and blue) a shading will be given.[2] They are:

$$s_R = \frac{R|\text{Cos }\theta|}{r + C} + M$$

$$s_G = \frac{G|\text{Cos }\theta|}{r + C} + M$$

$$s_B = \frac{B|\text{Cos }\theta|}{r + C} + M$$

Here s_R, s_G, and s_B, are the shading functions for the red, green, and blue color components of the polygon. The values of R, G, and B define the "color" of the polygon. What this means will now be explained.

The aspects of color include hue, intensity, and saturation. By setting the values of R, G, and B the three aspects of color may be specified.

The saturation of a color, in a sense, refers to the purity of the color. Pastel colors are unsaturated, brilliant colors are more saturated. The control of saturation with the formulas consists in adding or subtracting a constant to or from each of the three values R, G, and B. Adding a constant reduces saturation of the color. Subtracting a constant increases saturation. The maximum saturation under these circumstances occurs when one or more of R, G, and B has the value zero.

Hue is the aspect of color that is most often associated with the word "color". When one or more of R, G, and B are zero, then hue is

determined by the proportions of remaining values. If red and green are mixed additively then yellow results. If blue and green are mixed then cyan (sky blue) results. If red and blue are mixed then magenta (lavender) results. Variations in the ratios of these mixtures result in variations in hue.

The brightness or intensity of a color simply refers to the amount of colored light present. The terms "brightness" and "saturation" should never be confused since a color may be unsaturated but still be bright. In the color equations given above the magnitudes of R, G, and B, in a sense, control brightness. For example, if R and G have the value 1 and B equals 0, then the hue of yellow results. If R and G have the value 10 and B equals 0 then yellow still results but the yellow is ten times brighter.

Brightness is probably the most difficult parameter to judge and control. The problems arise due to the way idiomatic language refers to color. If red, yellow and blue patches of color are made so they reflect the same amount of light then disturbing results occur. The red patch may be described as "Red." If this is the case then the yellow patch will be described as "muddy brown" and the blue patch as "brilliant blue." The terms "red," "yellow" and "blue," used idiomatically, refer to entirely different brightness ranges. We can compensate for these differences in brightness by observing the percentages of reflectance from some "good" colors. If we do this for "magenta," "red," "yellow," "green," "cyan," and "blue," then percentages of reflectance are respectively: 20%, 20%, 70%, 15%, 20%, and 5%.

The worker can use these numbers as rough guides as to how he should assign values to R, G, and B.

From the above it might be expected that control of R, G, and B will give complete visual control of color. If, however, color pictures of objects are made with the above equations, the results obtained are disappointing. The pictures appear to have been made in black and white with each area painted the appropriate color. The colors do not seem to be part of the objects. Instead they appear to be masked onto the two-dimensional picture. (see Figure 5c)

These disappointing results are due to an oversight in constructing our formulas. The basic problem is due to the fact that light is reflected from objects in two ways. One component of reflected light is due to the specular properties of the objects. This light is not spectrally altered when reflected but instead has the same color properties as the source. The second component is the one modified by the color of the object. The functions that describe these components are generally quite different. The following three functions, although not physically sound, yield excellent results.

$$S_R = \frac{R \mid \text{Cos } \theta \mid^u + W \mid \text{Cos } \theta \mid^w}{r + C} + M$$

$$S_R = \frac{G \mid \text{Cos } \theta \mid^u + W \mid \text{Cos } \theta \mid^w}{r + C} + M$$

$$S_R = \frac{B \mid \text{Cos } \theta \mid^u + W \mid \text{Cos } \theta \mid^w}{r + C} + M$$

Here R, G, B, r, Θ, C and M are the same as in the previous equations. W changes the intensity of specularly reflected white light. The variables u and w correspond to the respective ways Θ affects the chromatic and specular reflectance. For example: If w is given a value of 10, then the specular component consists of sharp highlights on the object. This gives the illusion that the surface of the object is shiny. If smaller values are assigned to w, then a dull finish is approximated. The role of u is somewhat the same as w except that small values of u (1/2 to 1) have been found to yield good results. If pictures are made with these formulas, the results are quite spectacular. The coloring is vivid and real looking. (see Figure 5d and compare with Figure 5c) Although this is the case, modifications will be made to improve these formulas as new aspects of lighting and coloring become important.

This discussion has been included because the literature on color emphasizes analysis rather than synthesis. The emphasis of the existing literature makes it hard to discover what is important in constructing a shading and color model. In any case the control of these shading functions associated with each polygon allows the user to achieve a very rich assortment of lighting and reflectance qualities. Objects can be made to appear metallic and glossy, they can be brilliantly colored, or be of a very subtle hue. These capabilities combine to give the graphics user a latitude and freedom that he has not previously had in his ability to visually represent data.

CONCLUSION

This paper has dealt with some of the basic problems that arise in producing two-dimensional halftone images of objects described in three-space. Some new attitudes and techniques have been discussed. With these attitudes and techniques as a basis, it is natural to ask: What extensions to this kind of graphics are possible? Following is a list of potential extensions: The extension of the input to include curved surfaces as well as planar polygons; the inclusion of lines and text in the same image as halftone shading: the ability to specify transparency and translucency of objects as well as reflectivity; the ability to specify multiple light sources of various locations, intensities and colors; the ability to specify the filtering qualities of an atmosphere in order to simulate fog and haze. Each of these topics provide a research area in which no extensive work has been done.

I feel strongly that this enhancement of visual communication between machine and man will yield great advances in the effective utilization of computers. Data which once was represented as a sea of printed numbers can be represented as revealing visual patterns. Highly complex abstract relationships can be understood at a glance by their reduction to visual relationships. (See Figure 6) The development and perfection of this computation tool will allow man to more easily and effectively understand and solve the problems that confront him.

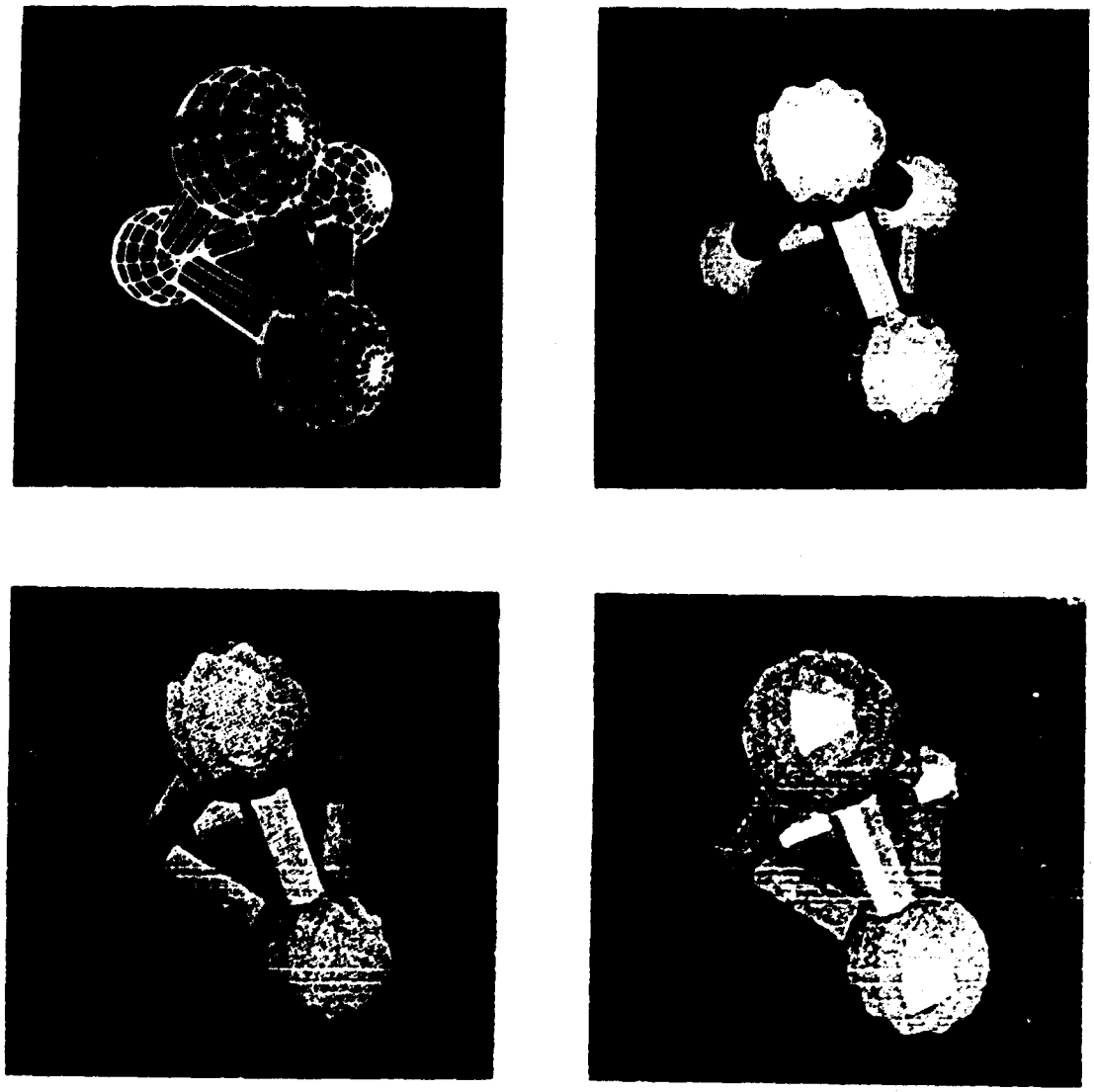


Fig. 5

These figures are representative of various kinds of shading rules. (a) displays only those points in the display file. This method yields a line drawing. (b) uses the display file along with shading functions associated with each polygon. (c) is a color representation that uses shading functions associated with each additive primary color. There are no specular components in this picture. (d) is the same as (c) except that specular components have been added to increase realism.

Fig. 6

These two figures represent different views of a two-dimensional surface in four-dimensional space. Here variation in hue on the surface represents variation in the fourth variable. Red indicates high values and green indicates low values. Orange, yellow, and yellow-green indicate intermediate values.

BIBLIOGRAPHY

1. Appel, A. "Some Techniques for Shading Machine Renderings of Solids, " *AFIPS Conference Proceedings, Spring Joint Computer Conference*, XXXII (1968), pp. 37-45.

2. Evans, Ralph M., et al. *Principles of Color Photography*. New York: John Wiley & Sons, Inc., 1953.

3. Floyd, Robert W. "Nondeterministic Algorithms," *Journal of the Association for Computing Machinery*, XIV, No. 4 (October, 1967), pp. 636-644.

4. Roberts, L.G. "Homogeneous Matrix Representation and Manipulation of N-Dimensional Constructs," Notes for English Summer Conference Course, University of Michigan, 1965.

5. Sutherland, I. E. "Computer Graphics-Ten Unsolved Problems," *DATAMATION*, XII, No. 5 (May, 1966), pp. 22-27.

6. Warnock, John E. *A Hidden Line Algorithm For Halftone Picture Representation*. Technical Report 4-5. Salt Lake City, Utah: University of Utah, 1968.

7. Weiss, R. A. "BE VISION, A Package of IBM 7090 FORTRAN Program to Draw Orthographic Views of Plane and Quadric Surfaces," *Journal of the Association for Computing Machinery*, XIII, No. 2 (February, 1966), pp. 194-204.

8. Wylie, C., et al. "Half-Tone Perspective Drawings by Computer," *AFIPS Conference Proceedings, Fall Joint Computer Conference*, XXXI (1967), pp. 49-58.